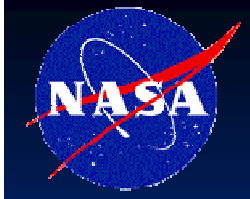
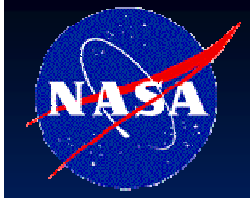


A Core Plug and Play Architecture for Component-Based Reusable Flight Software Systems

Jonathan Wilmot
Jonathan.J.Wilmot@nasa.gov
Maureen Bartholomew
Maureen.O.Bartholomew@nasa.gov

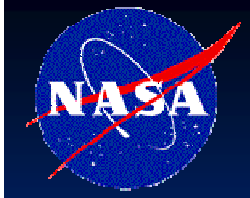
NASA GSFC Code 582



Mission Challenges

- Missions pushing for more spacecraft autonomy
- Increasing mission complexity
- Reduce cost, schedule, and risk
- Maintain flight software quality

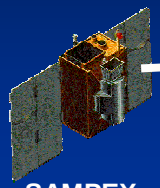
- GSFC approach
  - Core Flight Executive (cFE)
    - Platform-independent, service oriented flight software framework
  - Core Flight Software System (CFS)
    - "Configure and Play" catalog of reusable components
  - Integrated Development Environment (IDE)



Heritage Analysis

- Bring together senior software engineers with wide experiences
- Analyze previous missions architectures
- Extract common requirements
- Establish architecture goals for next generation with evolvability and expandability in mind
- Develop formal requirements

Heritage Mission Analysis



SAMPEX
(launched 8/92)

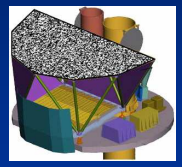
SWAS
(launched 12/98)

TRACE
(launched 3/98)

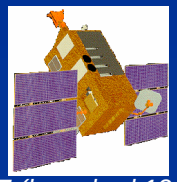
WIRE
(launched 2/99)

SMEX-Lite

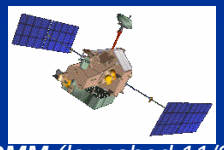
Triana
(waiting for launch)



Swift BAT
(12/04)

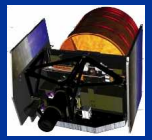


XTE *(launched 12/95)*

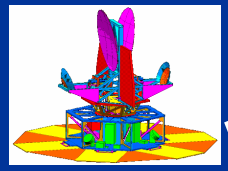


TRMM *(launched 11/97)*

JWST ISIM (2011)



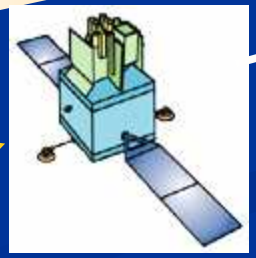
IceSat GLAS *(01/03)*



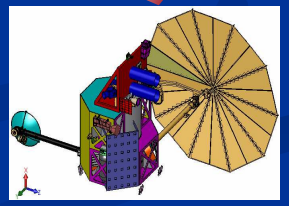
MAP *(launched 06/01)*

Evolved FSW Architecture

ST-5 *(5/06)*



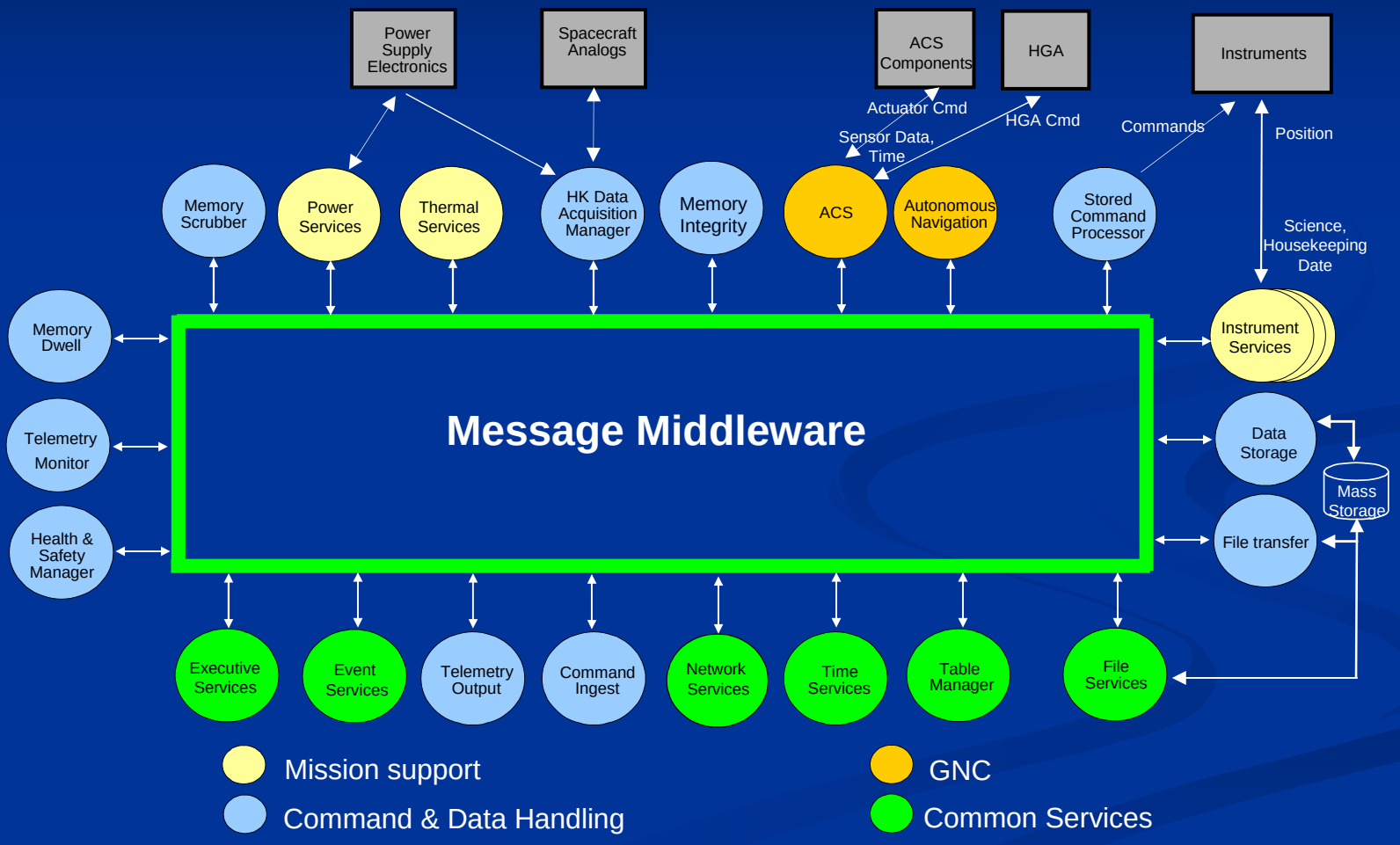
SDO (2007)

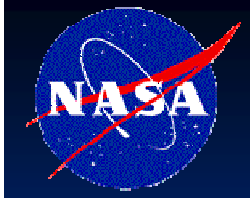


LRO (2009)

Future Spacecraft and Instruments

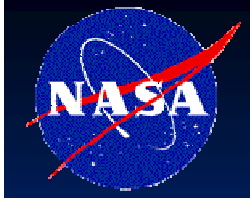
Heritage Architecture





Performance Design Goals

- Rad-Hard Spaceflight platforms are performance constrained
  - Small core software footprint
    - 512 kilobytes to 1 megabyte
  - Scaleable core memory utilization
    - 1 to 2 megabytes
  - Core processor utilization
    - Less than 5% over 1 second

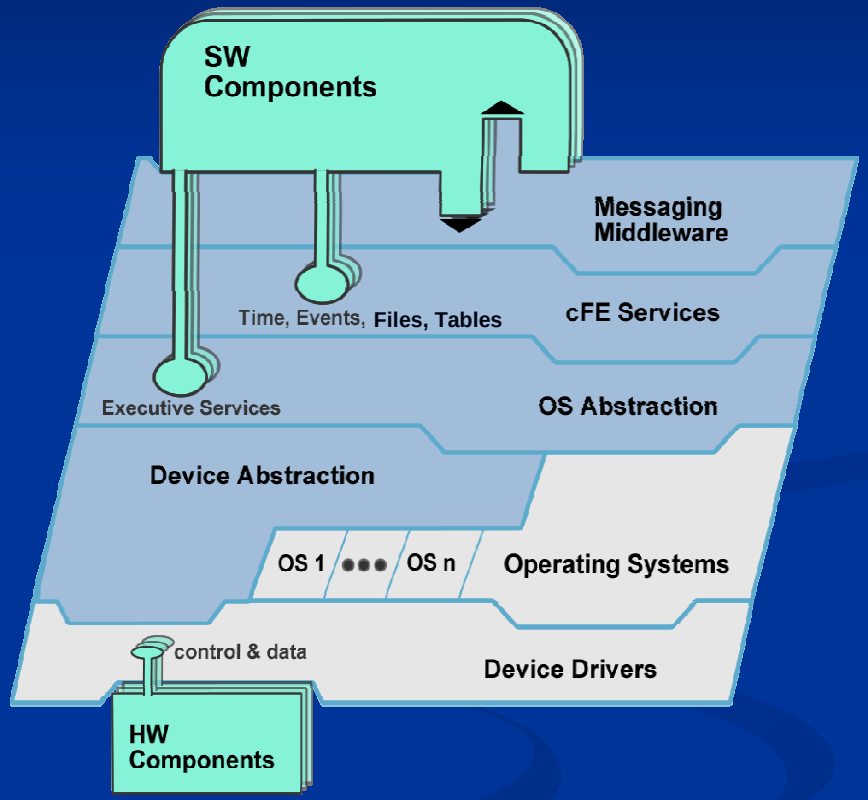


Run-Time Environment

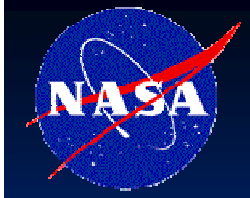
- Component Modularity
  - Software decomposed along clean functional lines
  - Components interact with system through API
  - Components can be individually compiled and linked
- Dynamic Loading/Unloading
  - Components can be loaded and unloaded from running system
  - Support both static and dynamic linking
  - Component cleanup
- Operational Transparency
  - Operator visibility into the current running state
  - System utilization monitors
  - Built in diagnostics

Implementation

- Layered Architecture
- Publish/Subscribe Middleware/Bus
- Standard Application Programmer Interface
- Run-time Services

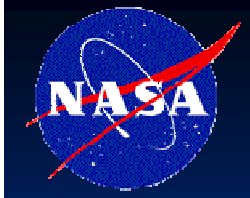


Core Flight Executive (cFE)



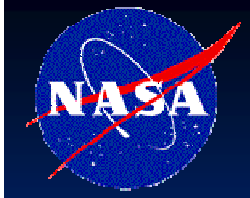
cFE Services

- OS Abstraction
  - Normalized API and services for common flight operating systems
- Executive Services
  - System startup/restart
  - Component start/stop/load/unload
  - Exception and interrupt handling
  - System logs
  - Performance monitoring utilities



cFE Services

- Software Bus
  - Publish and subscribe messaging middleware
  - Local and networked inter-component messaging
  - One-to-one, one-to-many, many-to-one
  - Poll, Pend, and Quality of Service
- Event Handler
  - Event formatting and filtering
  - Local logging
  - Event port selection

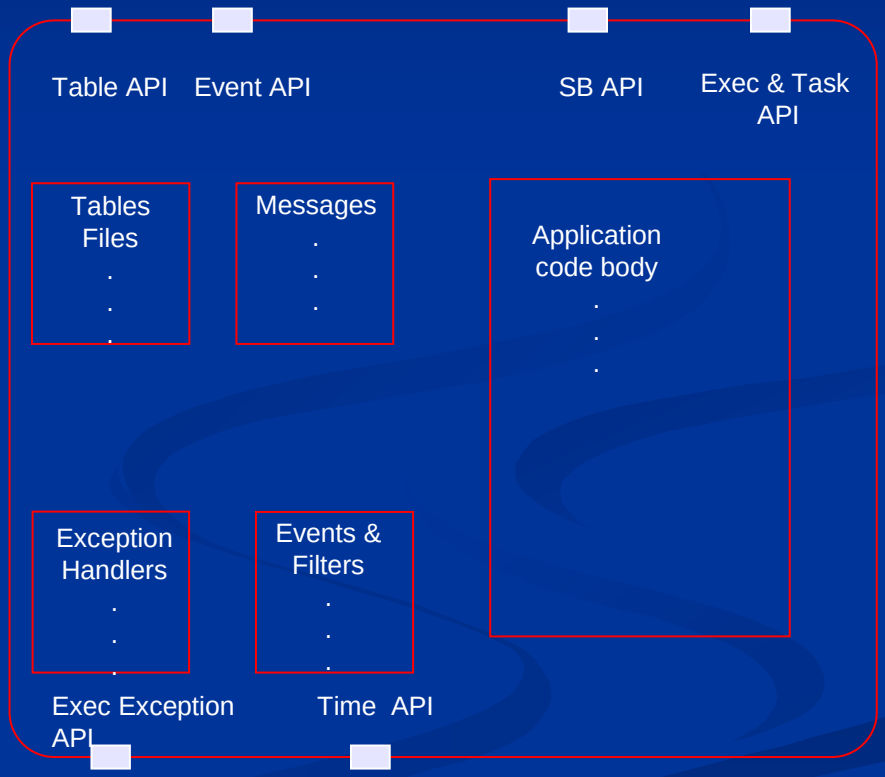


cFE Services

- Table Management
  - Named data storage
  - Load/Dump/Activate
- Time Management
  - Time maintenance and external interfaces
  - Local distribution
  - Time utilities
- File Services
  - Common Headers
  - Utilities

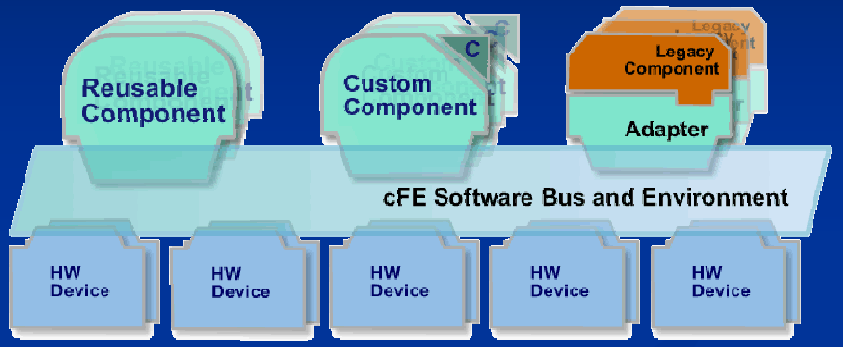
Software Component Example

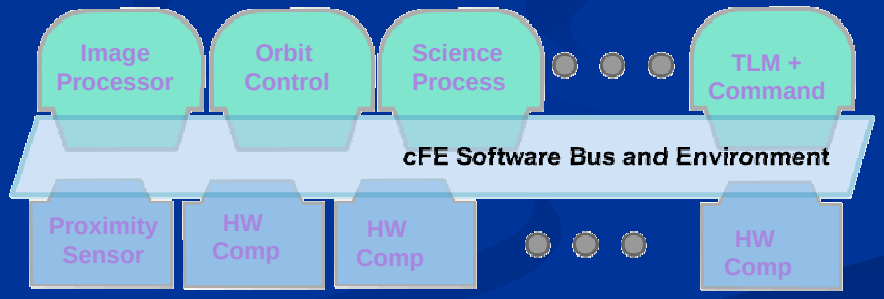
- Interface only through five cFE API's.
- cFE services track component resources
  - Capability to clean up after component faults
  - Provides utilization statistics
- A components contains all data needed to define it's operation.
- Components register for services
  - Register exception handlers
  - Register Event counters and filter
  - Register Tables
  - Publish messages
  - Subscribe to messages
- Components include artifacts, Requirements, Unit tests, build tests, documentation, & code



Plug and Play
(Configure and Play)

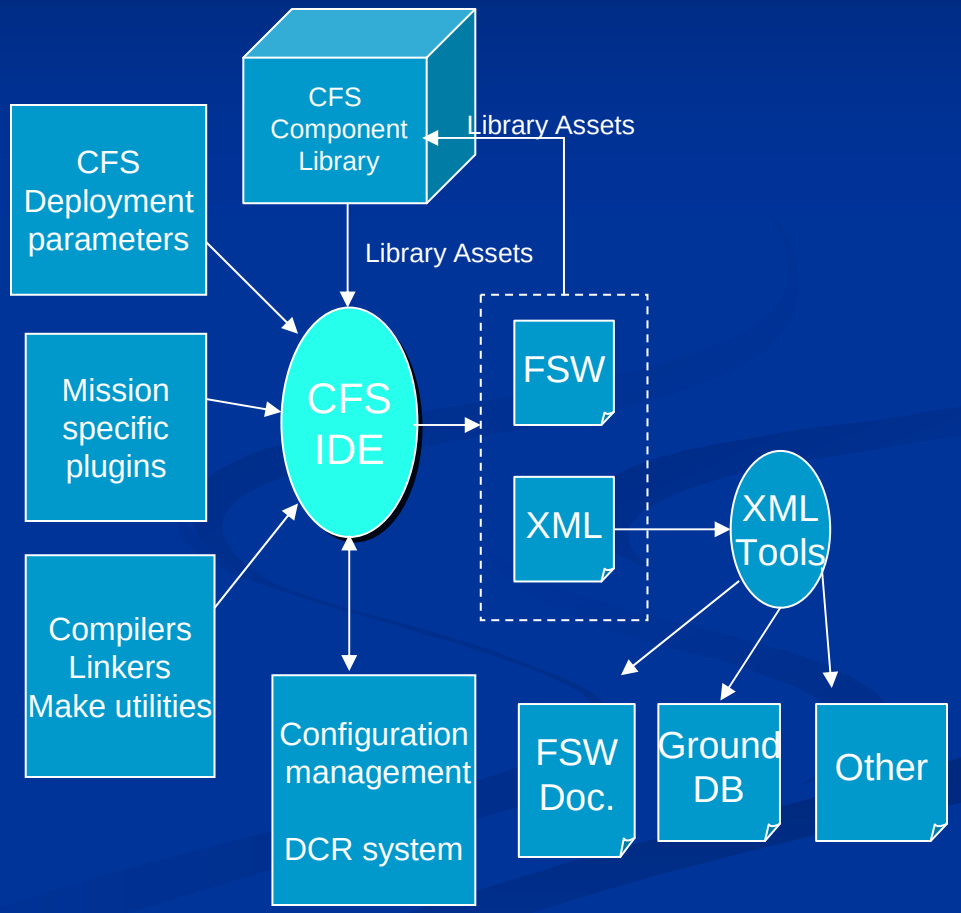
- Common FSW functionality has been abstracted into a catalog of **reusable components and services**.
- **Tested, Certified, Documented**
- **A system is built from:**
  - Core services
  - Reusable components
  - Custom mission specific components
  - Adapted legacy components
- Software components can be configured, built and plugged at any time during development or in flight.
- Software components can be associated with plug in devices or system modes and be loaded as needed

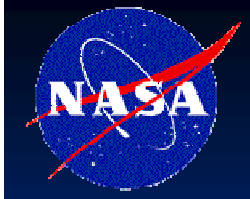




Integrated Development Environment (IDE)

- The CFS IDE is a growing set of integrated tools that support the management, development and configuration of a CFS deployment
- Tools are hosted on the open source Eclipse platform being adopted by many tool vendors, including Wind Rivers' VxWorks
- IDE supports rapid deployment using pick and click graphical user interface for system configuration
- CFS Component consists of: Requirements, Design, Code, unit & build tests, User's guide, and XML description of cmd/tlm database





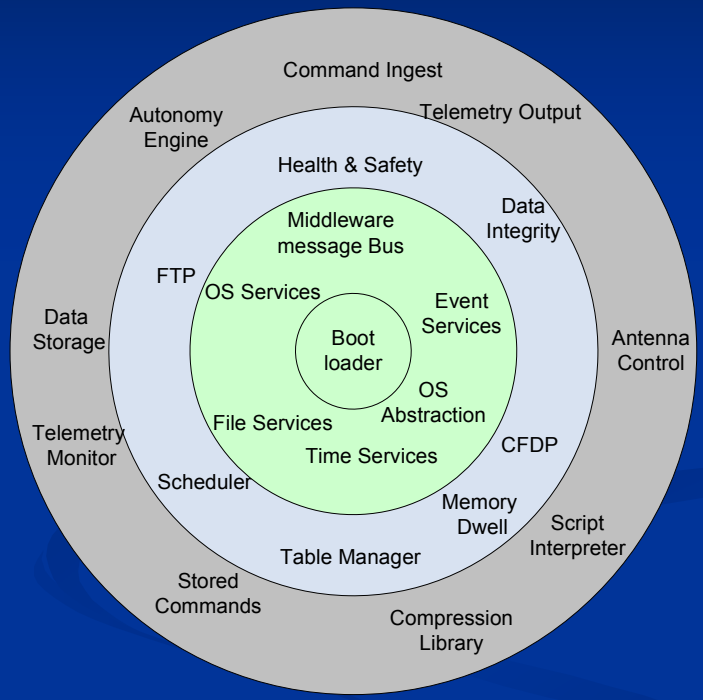
Target Platforms

- **Embedded Systems**
  - PPC/mcp750 running VxWorks 5.5, VxWorks 6.x, Linux
  - PPC/mcp405 running Linux
  - BAE RAD750
- **Desktop Systems**
  - PPC/X86 Macintosh running OS X
  - X86/PC running Linux
  - X86/PC running Cygwin under Windows

Future targets: Coldfire running RTEMS, PPC/mcp750 running RTEMS and LynxOS

Platform Example

- Interchangeable avionics boxes have a core ring of standard services and functions.
  - Actual code and implementation can be different for each vendor, but need to meet the API's and ICD's
  - Messages to alert operators or other components of asynchronous events
  - Common software load/dump interfaces
  - Common set of file service interfaces
  - Core ring can use memory management unit (MMU) for self protection
  - Network enabled Boot loader in hardened ROM
- Benefits
  - Speed prototyping and integration times
  - Common training for maintenance teams.
  - Reduced cost for FSW development and maintenance
  - Boxes can take on "personalities" as needed
  - Potential for Fewer flight spares



Status

- **2004 multi-CPU/Box prototype demonstrated**
  - Core, generic FSW services and Software components
  - Dynamic application load and startup
  - Dynamic message bus reconfiguration for "Box faults"
- **2005 Version 3.1 cFE, delivered to Autonomy Test Bed (ATB), LRO Nov 16 2005**
  - VxWorks 5.4, Static linking, S-rec file startup
  - For ATB effort, cfE has worked with Vxworks, OSX, and Linux
  - Unit Test Framework (UTF) developed for white box component tests
  - **December CHIPS on orbit demonstration**
    - VxWorks 5.4
    - IP/UDP command and telemetry
    - cFE loaded onto CHIPS providing testbed for automation components
- **2006 Lunar Reconnaissance Orbiter (LRO) configuration baseline cFE 4.2**
  - VxWorks 6.2, Static linking, ELF file startup
  - IP/UDP command and telemetry
- **In process of obtaining open source release of core Flight Executive software**